Mental Health and Access to Care Survey 2022: ETL and Data Visualization

INTRODUCTION :

This analysis utilizes data from the Mental Health and Access to Care Survey (MHACS) 2022 Public Use Microdata File (PUMF), provided by Statistics Canada. The dataset offers a comprehensive view of self-rated mental health, life satisfaction and perceived stress, segmented by age groups, gender and marital status. The objective is to extract, transform and analyse this data, uncovering key patterns and trends through detailed visualisations.

The data, stored in CSV format within the Databricks File System (DBFS), will be loaded into a Spark DataFrame for ETL (Extract, Transform, Load) operations, followed by in-depth analysis and visualization.

DATA LOADING :

The CSV file containing the MHACS 2022 data is stored in the Databricks File System (DBFS). The file is initially loaded into a Spark DataFrame without type inference, meaning all data is treated as strings. This step ensures that the data is correctly imported for further processing. A visual inspection of the DataFrame is done to confirm that the data is loaded properly and is ready for transformation.

```
# Extracting Data from MHACS Survey File stored in Databricks File System (DBFS)
# File location and type
file_location = "/FileStore/tables/pumf-2.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Table

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9
1	PUMFID	GEODVPSZ	DHHGMS	DHHGAGE	GENDER	GEN_01	GEN_02A	GEN_08B	GEN_08C	GEN_02A1
2	615402.0	3.0	1.0	7.0	1.0	2.0	3.0	2.0	3.0	3.0
3	615403.0	4.0	3.0	2.0	2.0	4.0	3.0	4.0	4.0	3.0
4	615404.0	4.0	1.0	6.0	1.0	2.0	3.0	3.0	3.0	2.0
5	615405.0	4.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	1.0
6	615406.0	4.0	3.0	2.0	1.0	2.0	2.0	2.0	1.0	2.0
7	615407.0	2.0	3.0	3.0	1.0	1.0	3.0	2.0	3.0	1.0
8	615408.0	4.0	1.0	3.0	1.0	3.0	2.0	3.0	4.0	5.0
9	615409.0	4.0	3.0	1.0	2.0	3.0	1.0	3.0	1.0	5.0
10	615410.0	4.0	1.0	6.0	1.0	3.0	3.0	2.0	3.0	1.0
11	615411.0	4.0	4.0	6.0	2.0	4.0	4.0	4.0	4.0	4.0
12	615412.0	4.0	3.0	4.0	1.0	1.0	3.0	1.0	4.0	4.0
13	615413.0	4.0	3.0	1.0	1.0	1.0	3.0	2.0	3.0	2.0
14	615414.0	4.0	1.0	6.0	1.0	2.0	3.0	1.0	3.0	1.0
15	615415.0	4.0	4.0	8.0	2.0	2.0	3.0	3.0	4.0	3.0

556+ rows | Truncated data due to byte limit

DATA CLEANING AND TRANSFORMATION :

Reading Data with Headers: The data is re-read with the first row interpreted as headers, allowing column names to be assigned correctly. This step makes the dataset more manageable and easier to work with for subsequent transformations.

Column Selection and Type Casting: Specific columns relevant to the analysis are selected from the dataset. These columns include identifiers like PUMFID, demographic details such as age, gender, and marital status and various mental health indicators. The dataset is limited to the first 1000 rows to streamline the analysis. Additionally, certain columns are cast to integer types to facilitate numerical analysis, replacing their initial string formats.

Filtering Invalid Data: The data is further cleaned by removing rows that contain invalid or irrelevant values in key columns. This step is crucial for ensuring the accuracy and reliability of the subsequent analysis. For example, rows with values indicating "Don't Know", "Refusal" or "Not Stated" in the mental health-related columns are filtered out.

Categorical Transformation: New categorical columns are created based on the existing numerical codes in specific columns. This transformation involves mapping numerical codes to meaningful labels, making the data more interpretable. For instance, marital status codes are translated into categories like "Married", "Living common law" or "Never married." Similar transformations are applied to age groups and gender codes.

```
# Read the data with the first row as header
df = spark.read.option("header", "true").csv("/FileStore/tables/pumf-2.csv")

# Display the entire dataframe
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Read data into a dataframe
df = spark.read.format("csv").option("header", "true").load("/FileStore/tables/pumf-2.csv")
display(df)
```

After executing this code, the CSV file from Statistics Canada located at /FileStore/tables/pumf-2.csv is successfully read into a Spark DataFrame with the first row interpreted as headers. The display(df) command provides a visual inspection of the DataFrame, ensuring that the data is correctly loaded with appropriate column names and is ready for further transformation and analysis.

The process involves further refining the data. After loading the CSV data into a Spark DataFrame, we need to select specific columns, limit the dataset to the first 1000 rows and ensure that certain columns are cast to the appropriate data types for accurate analysis

```
from pyspark.sql.functions import col

# Select the desired columns
df_selected = df.select("PUMFID", "DHHGMS","DHHGAGE", "GENDER", "GEN_01", "GEN_02B", "GEN_07", "GEN_09", "SCRDMEN", "CEX_0

# Keep the first 1000 rows
df_selected = df_selected.limit(1000)

df_selected = df_selected.withColumn("SCRDMEN", col("SCRDMEN").cast("integer"))
df_selected = df_selected.withColumn("GEN_01", col("GEN_01").cast("integer"))
df_selected = df_selected.withColumn("GEN_02B", col("GEN_02B").cast("integer"))
df_selected = df_selected.withColumn("GEN_07", col("GEN_07").cast("integer"))
df_selected = df_selected.withColumn("GEN_09", col("GEN_09").cast("integer"))

# Display the resulting dataframe
display(df_selected)
```

After executing this code, the DataFrame df_selected will contain only the selected columns: "PUMFID", "DHHGMS", "DHHGAGE", "GENDER", "GEN_01", "GEN_02B", "GEN_07", "GEN_09", "SCRDMEN" and "CEX_05", limited to the first 1000 rows of the dataset. The specified columns ("SCRDMEN", "GEN_01", "GEN_02B", "GEN_07" and "GEN_09") are changed to integer type from string data type to facilitate numerical analysis. The display(df_selected) command provides a visual inspection of the refined DataFrame, ensuring the data is ready for further transformation and visualization tasks.

After that, the next step is to remove rows with invalid or irrelevant values in specified columns. This data cleaning step ensures that the dataset is accurate and reliable for subsequent analysis.

```
# Remove rows where GEN_01 has values 7 or 8
df_selected = df_selected.filter((df_selected['GEN_01'] != 7) & (df_selected['GEN_01'] != 8))

# Remove rows where GEN_02B has values 97, 98, or 99
df_selected = df_selected.filter((df_selected['GEN_02B'] != 97) & (df_selected['GEN_02B'] != 98) & (df_selected['GEN_02B'] != 99))

# Remove rows where GEN_07 has values 7, 8, or 9
df_selected = df_selected.filter((df_selected['GEN_07'] != 7) & (df_selected['GEN_07'] != 8) & (df_selected['GEN_07'] != 9))

# Remove rows where GEN_09 has values 6, 7, 8, or 9
df_selected = df_selected.filter((df_selected['GEN_09'] != 6) & (df_selected['GEN_09'] != 7) & (df_selected['GEN_09'] != 8) & (df_selected['GEN_09'] != 9))

# Remove rows where SCRDMEN has value 9
df_selected = df_selected.filter(df_selected['SCRDMEN'] != 9)

display(df_selected)
```

After executing this code, the DataFrame df_selected will have rows removed based on specific conditions for columns "GEN_01", "GEN_02B", "GEN_07", "GEN_09" and "SCRDMEN". Specifically:

- Rows where "GEN_01" has values 7 (Don't Know) or 8 (Refusal) are removed.
- Rows where "GEN_02B" has values 97 (Don't Know), 98 (Refusal) or 99 (Not Stated) are removed.
- Rows where "GEN_07" has values 7 (Don't Know), 8 (Refusal) or 9 (Not Stated) are removed.
- Rows where "GEN_09" has values 6 (Valid Skip), 7 (Don't Know), 8 (Refusal) or 9 (Not Stated) are removed.
- Rows where "SCRDMEN" has a value of 9 (Not Stated) are removed.

The display(df_selected) command provides a visual inspection of the cleaned DataFrame, ensuring that the data is now in a suitable state for further analysis and visualization tasks.

After cleaning the data, the next step is to create new categorical columns based on existing numerical codes in specific columns. This step converts coded survey responses into meaningful categorical labels, making the data more interpretable for analysis and visualization.

```
# Define conditions and corresponding values for new column
condition_col1 = df_selected['DHHGMS'] == 1.0
condition_col2 = df_selected['DHHGMS'] == 2.0
condition_col3 = df_selected['DHHGMS'] == 3.0
condition_col4 = df_selected['DHHGMS'] == 4.0
condition_col5 = df_selected['DHHGMS'] == 5.0
condition_col99 = df_selected['DHHGMS'] == 99.0

value_col1 = 'Married'
value_col2 = 'Living common law'
value_col3 = 'Never married'
value_col4 = 'Separated or Divorced'
value_col5 = 'Widowed'
value_col99 = 'Not stated'

# Insert values into new column based on conditions
df_selected = df_selected.withColumn('DHHGMS_Category',
                                     when(condition_col1, value_col1)
                                     .when(condition_col2, value_col2)
                                     .when(condition_col3, value_col3)
                                     .when(condition_col4, value_col4)
                                     .when(condition_col5, value_col5)
                                     .when(condition_col99, value_col99)
                                     .otherwise(None))
```

```
# Define conditions and corresponding values for DHHGAGE_Category
condition_col1 = df_selected['DHHGAGE'] == 1.0
condition_col2 = df_selected['DHHGAGE'] == 2.0
condition_col3 = df_selected['DHHGAGE'] == 3.0
condition_col4 = df_selected['DHHGAGE'] == 4.0
condition_col5 = df_selected['DHHGAGE'] == 5.0
condition_col6 = df_selected['DHHGAGE'] == 6.0
condition_col7 = df_selected['DHHGAGE'] == 7.0
condition_col8 = df_selected['DHHGAGE'] == 8.0

value_col1 = '15 to 19 years'
value_col2 = '20 to 24 years'
value_col3 = '25 to 29 years'
value_col4 = '30 to 34 years'
value_col5 = '35 to 44 years'
value_col6 = '45 to 54 years'
value_col7 = '55 to 64 years'
value_col8 = '65 years or older'
```

```
# Insert values into DHHGAGE_Category based on conditions
df_selected = df_selected.withColumn('DHHGAGE_Category',
                                     when(condition_col1, value_col1)
                                     .when(condition_col2, value_col2)
                                     .when(condition_col3, value_col3)
                                     .when(condition_col4, value_col4)
                                     .when(condition_col5, value_col5)
                                     .when(condition_col6, value_col6)
                                     .when(condition_col7, value_col7)
                                     .when(condition_col8, value_col8)
                                     .otherwise(None))

# Define conditions and corresponding values for GENDER_Category
condition_col1 = df_selected['GENDER'] == 1.0
condition_col2 = df_selected['GENDER'] == 2.0
condition_col9 = df_selected['GENDER'] == 9.0

value_col1 = 'Men'
value_col2 = 'Women'
value_col9 = 'Not Stated'

# Insert values into GENDER_Category based on conditions
df_selected = df_selected.withColumn('GENDER_Category',
                                     when(condition_col1, value_col1)
                                     .when(condition_col2, value_col2)
                                     .when(condition_col9, value_col9)
                                     .otherwise(None))
```

```
# Define conditions and corresponding values for CEX_05_Category
condition_col1 = df_selected['CEX_05'] == 1.0
condition_col2 = df_selected['CEX_05'] == 2.0
condition_col3 = df_selected['CEX_05'] == 3.0
condition_col4 = df_selected['CEX_05'] == 4.0
condition_col5 = df_selected['CEX_05'] == 5.0
condition_col6 = df_selected['CEX_05'] == 6.0
condition_col7 = df_selected['CEX_05'] == 7.0
condition_col8 = df_selected['CEX_05'] == 8.0
condition_col9 = df_selected['CEX_05'] == 9.0

value_col1 = 'Never'
value_col2 = '1 or 2 times'
value_col3 = '3 to 5 times'
value_col4 = '6 to 10 times'
value_col5 = 'More than 10 times'
value_col6 = 'Valid skip'
value_col7 = "Don't know"
value_col8 = 'Refusal'
value_col9 = 'Not stated'

# Insert values into CEX_05_Category based on conditions
df_selected = df_selected.withColumn('CEX_05_Category',
                                     when(condition_col1, value_col1)
                                     .when(condition_col2, value_col2)
                                     .when(condition_col3, value_col3)
                                     .when(condition_col4, value_col4)
                                     .when(condition_col5, value_col5)
                                     .when(condition_col6, value_col6)
                                     .when(condition_col7, value_col7)
                                     .when(condition_col8, value_col8)
                                     .when(condition_col9, value_col9)
                                     .otherwise(None))


display(df_selected)
```

Table | Self Rated Mental Health by Different Age Groups | Average Ratings by Gender | Self Perceived Health by Gende ···

	PUMFID	DHHGMS	DHHGAGE	GENDER	GEN_01	GEN_02B	GEN_07	GEN
1	615402.0	1.0	7.0	1.0	2	8	3	
2	615403.0	3.0	2.0	2.0	4	8	4	
3	615404.0	1.0	6.0	1.0	2	7	3	
4	615405.0	3.0	2.0	2.0	3	6	4	
5	615406.0	3.0	2.0	1.0	2	7	3	
6	615407.0	3.0	3.0	1.0	1	9	3	
7	615408.0	1.0	3.0	1.0	3	7	3	
8	615409.0	3.0	1.0	2.0	3	7	2	
9	615410.0	1.0	6.0	1.0	3	8	2	
10	615411.0	4.0	6.0	2.0	4	6	5	
11	615412.0	3.0	4.0	1.0	1	9	3	
12	615413.0	3.0	1.0	1.0	1	9	2	
13	615414.0	1.0	6.0	1.0	2	9	3	
14	615417.0	1.0	7.0	1.0	3	7	3	
15	615420.0	1.0	6.0	1.0	2	5	4	

580 rows

After executing this code, the DataFrame df_selected will have new categorical columns based on the specified conditions:

- DHHGMS_Category: Describes the marital status with labels such as 'Married', 'Living common law', 'Never married' etc.

- DHHGAGE_Category: Categorizes age groups with labels like '15 to 19 years', '20 to 24 years', '25 to 29 years' etc.

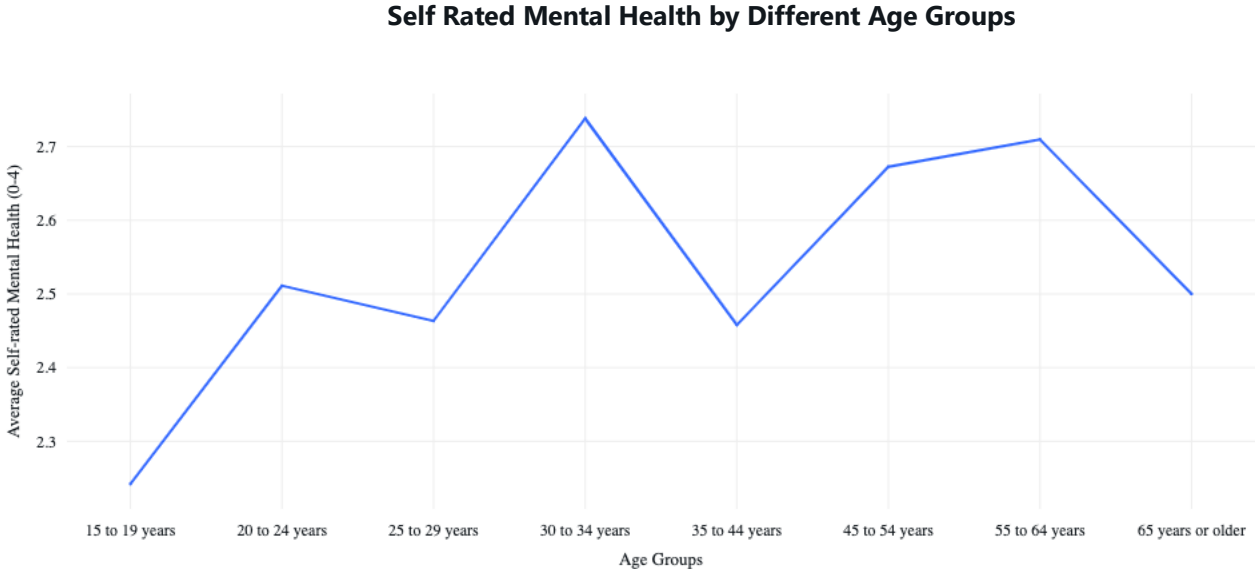
- GENDER_Category: Specifies gender with labels 'Men', 'Women' and 'Not Stated'.

- CEX_05_Category: Indicates frequency of certain behaviors with labels such as 'Never', '1 or 2 times', '3 to 5 times' etc.

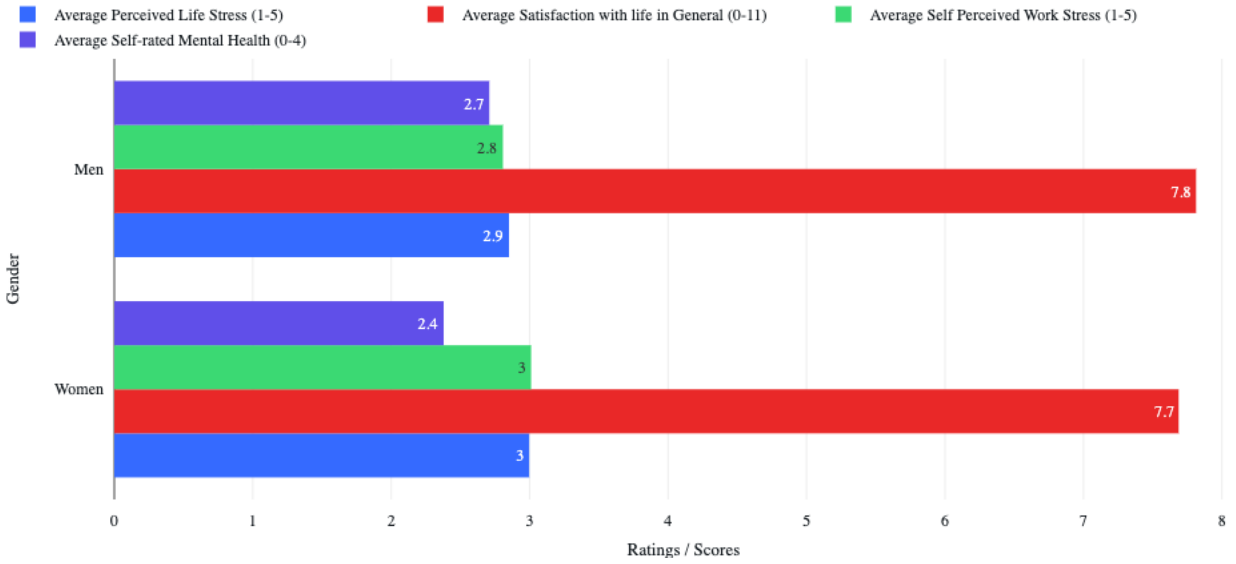
The display(df_selected) command provides a visual inspection of the enhanced DataFrame, ensuring that the data is now more interpretable and ready for further analysis and visualization tasks.

DATA VISUALISATION :

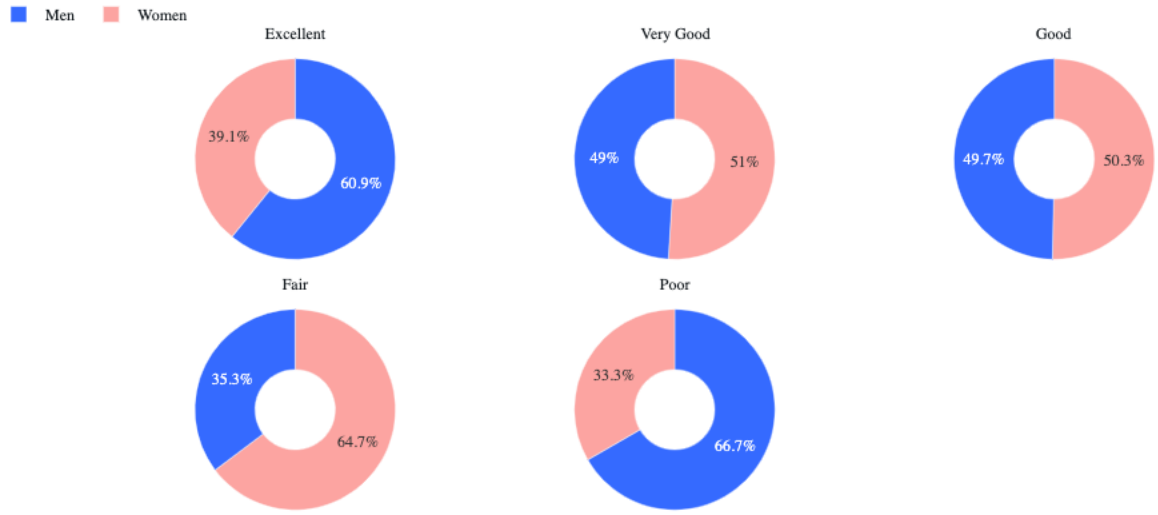
After successfully cleaning the data, the next step is to visualize various graphs. This involves creating visual representations of the data to uncover patterns, trends and insights that are crucial for analysis and decision-making.



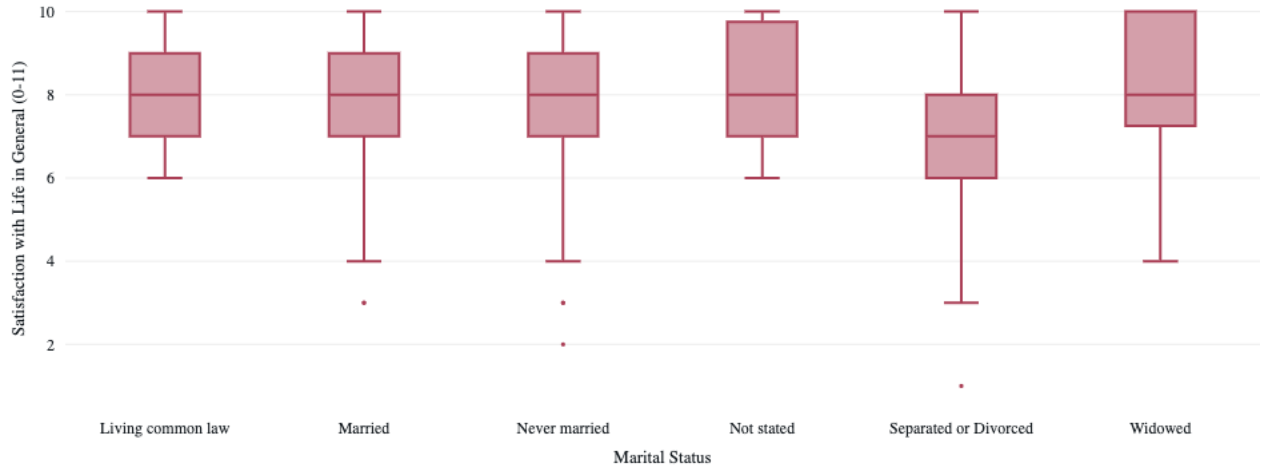
Average Ratings by Gender



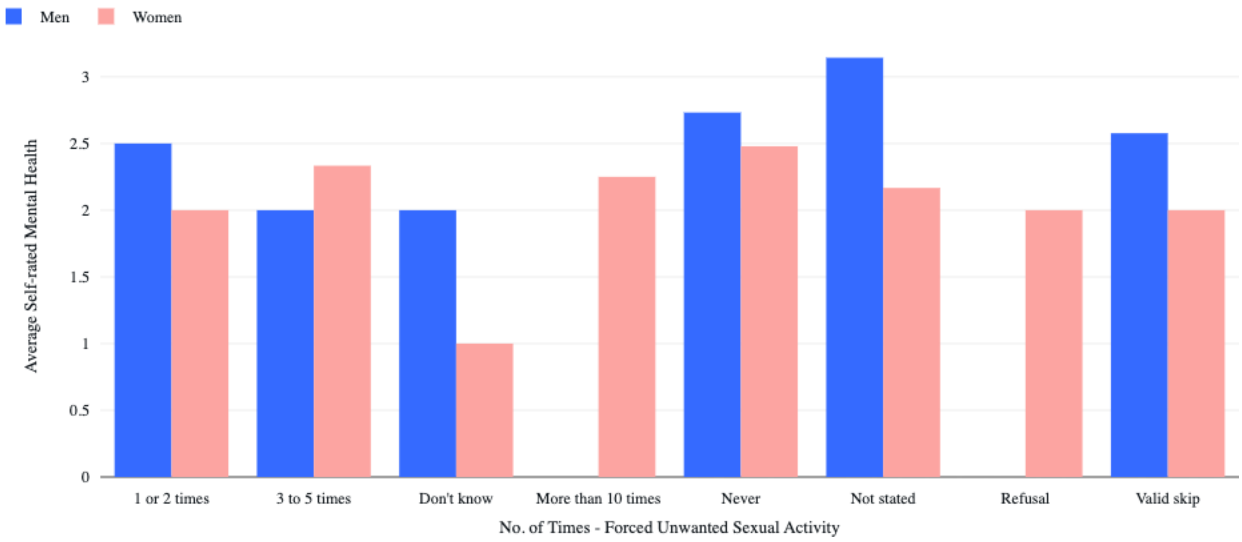
Self Perceived Health by Gender



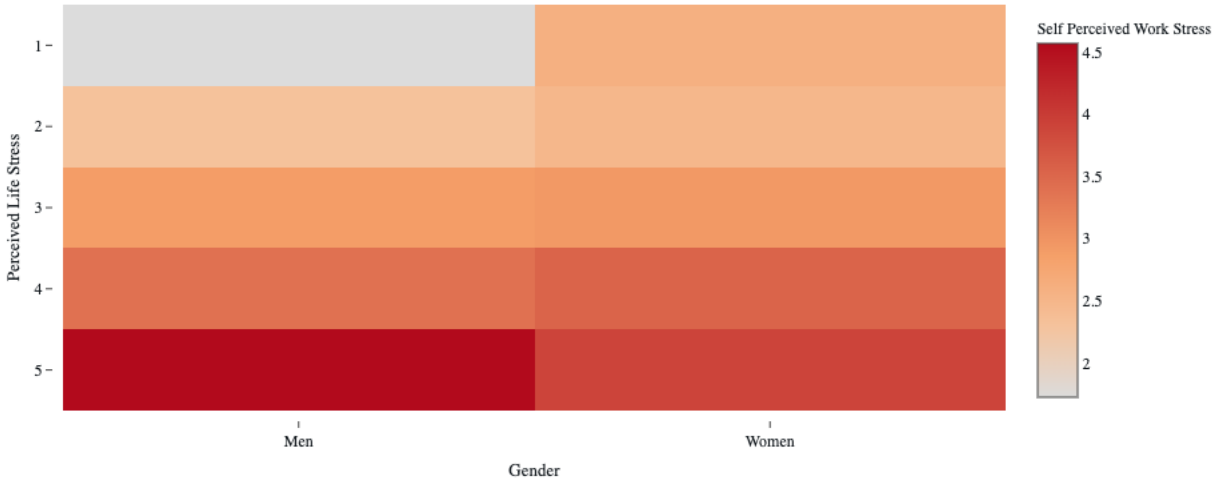
Satisfaction with Life by Marital Status



Self-Rated Mental Health by Frequency of Unwanted Sexual Activity and Gender



Perceived Life and Self-Perceived Work Stress Between Gender



Brief Insights:

- Self Rated Mental Health by Age Groups:

Self-rated mental health scores fluctuate across all age groups, with the highest average score observed in the 30-34 years age group and the lowest in the 15-19 years age group.

- Average Ratings by Gender:

Males and females show similar average scores for perceived life stress (1-5), satisfaction with life in general (0-11), self-perceived work stress (1-5) and self-rated mental health (0-4). Females tend to rate slightly lower in satisfaction with life in general and self-rated mental health compared to males.

- Self Perceived Health by Gender:

Among men, a larger percentage rate their health as "Excellent" compared to women, while women more frequently rate their health as "Fair" compared to men.

- Satisfaction with Life by Marital Status:

Separated or divorced individuals have the lowest median satisfaction with life in general (0-11) compared to other marital statuses.

- Self Rated Mental Health by Frequency of Unwanted Sexual Activity and Gender:

Women who have experienced unwanted sexual activity and don't know the number of times tend to report lowest self-rated mental health, which is lower than men, who report lowest mental health after experiencing such activity in both categories :three to five times and don't know.

- Perceived Life Stress vs. Self Perceived Work Stress by Gender:

Men with higher self-perceived work stress also tend to report higher perceived life stress compared to women, indicating a potential correlation between work stress and overall life stress.

CONCLUSION :

The analysis of the Mental Health and Access to Care Survey 2022 reveals significant insights into how self-rated mental health, life satisfaction and perceived stress vary across different demographic groups. Age plays a crucial role in self-rated mental health, with younger individuals reporting lower scores. Gender differences are evident, particularly in life satisfaction and self-perceived health, where men tend to rate their health more favorably than women. Marital status also influences life satisfaction, with separated or divorced individuals experiencing lower median scores. Additionally, the impact of unwanted sexual activity on mental health is pronounced, especially among women. Finally, a correlation between self-perceived work stress and overall life stress is observed, particularly in men.

These findings highlight the importance of targeted mental health support and interventions that consider age, gender, marital status and experiences of trauma to improve overall well-being.